# **[Android的Socket通信](http://www.cnblogs.com/zhujiabin/p/5675716.html)**

# 一、Socket通信简介

Android与服务器的通信方式主要有两种，一是Http通信，一是Socket通信。两者的最大差异在于，http连接使用的是“请求—响应方式”，即在请求时建立连接通道，当客户端向服务器发送请求后，服务器端才能向客户端返回数据。而Socket通信则是在双方建立起连接后就可以直接进行数据的传输，在连接时可实现信息的主动推送，而不需要每次由客户端想服务器发送请求。那么，什么是socket？Socket又称套接字，在程序内部提供了与外界通信的端口，即端口通信。通过建立socket连接，可为通信双方的数据传输传提供通道。socket的主要特点有数据丢失率低，使用简单且易于移植。

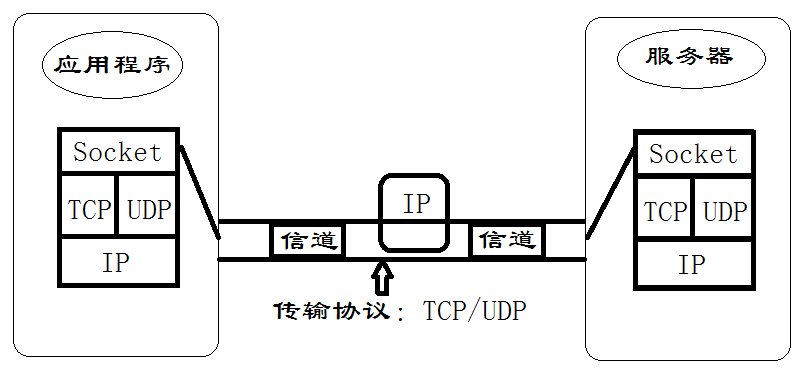
## 1.1什么是Socket Socket

是一种抽象层，应用程序通过它来发送和接收数据，使用Socket可以将应用程序添加到网络中，与处于同一网络中的其他应用程序进行通信。简单来说，Socket提供了程序内部与外界通信的端口并为通信双方的提供了数据传输通道。

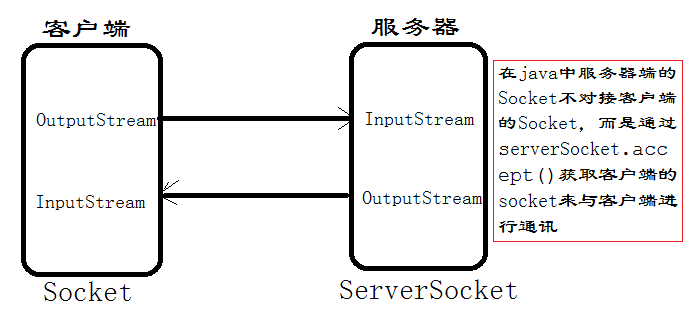
## 1.2Socket的分类

根据不同的的底层协议，Socket的实现是多样化的。本指南中只介绍TCP/IP协议族的内容，在这个协议族当中主要的Socket类型为流套接字（streamsocket）和数据报套接字(datagramsocket)。流套接字将TCP作为其端对端协议，提供了一个可信赖的字节流服务。数据报套接字使用UDP协议，提供数据打包发送服务。 下面，我们来认识一下这两种Socket类型的基本实现模型。

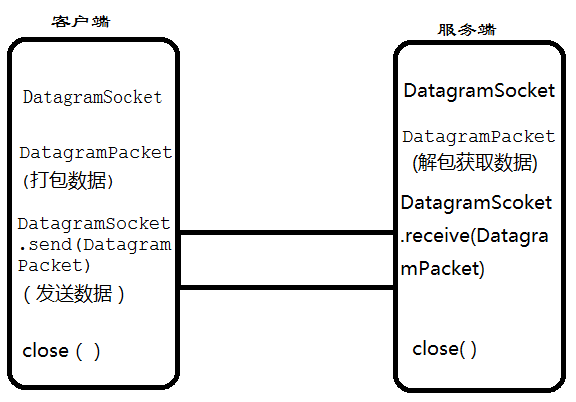
# 二、Socket 基本通信模型



## 2.1、TCP通讯模型



## 2.2、UDP通讯模型



# 三、Socket基本实现原理

## 3.1基于TCP协议的Socket

服务器端首先声明一个ServerSocket对象并且指定端口号，然后调用Serversocket的accept（）方法接收客户端的数据。accept（）方法在没有数据进行接收的处于堵塞状态。（Socketsocket=serversocket.accept()）,一旦接收到数据，通过inputstream读取接收的数据。

客户端创建一个Socket对象，指定服务器端的ip地址和端口号（Socketsocket=newSocket("172.168.10.108",8080);）,通过inputstream读取数据，获取服务器发出的数据（OutputStreamoutputstream=socket.getOutputStream()），最后将要发送的数据写入到outputstream即可进行TCP协议的socket数据传输。

## 3.2基于UDP协议的数据传输

服务器端首先创建一个DatagramSocket对象，并且指点监听的端口。接下来创建一个空的DatagramSocket对象用于接收数据（bytedata[]=newbyte[1024;]DatagramSocketpacket=newDatagramSocket（data，data.length））,使用DatagramSocket的receive方法接收客户端发送的数据，receive（）与serversocket的accepet（）类似，在没有数据进行接收的处于堵塞状态。

客户端也创建个DatagramSocket对象，并且指点监听的端口。接下来创建一个InetAddress对象，这个对象类似与一个网络的发送地址（InetAddressserveraddress=InetAddress.getByName（"172.168.1.120"））.定义要发送的一个字符串，创建一个DatagramPacket对象，并制定要讲这个数据报包发送到网络的那个地址以及端口号，最后使用DatagramSocket的对象的send（）发送数据。\*（Stringstr="hello";bytedata[]=str.getByte();DatagramPacketpacket=new DatagramPacket(data,data.length,serveraddress,4567);socket.send(packet);）

# 四、android 实现socket简单通信

## 前言：添加权限

<!--允许应用程序改变网络状态-->

<uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE"/>

<!--允许应用程序改变WIFI连接状态-->

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>

<!--允许应用程序访问有关的网络信息-->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<!--允许应用程序访问WIFI网卡的网络信息-->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>

<!--允许应用程序完全使用网络-->

<uses-permission android:name="android.permission.INTERNET"/>

## 4.1使用TCP协议通信

android端实现：

protected void connectServerWithTCPSocket() {

Socket socket;

try {// 创建一个Socket对象，并指定服务端的IP及端口号

socket = new Socket("192.168.1.32", 1989);

// 创建一个InputStream用户读取要发送的文件。

InputStream inputStream = new FileInputStream("e://a.txt");

// 获取Socket的OutputStream对象用于发送数据。

OutputStream outputStream = socket.getOutputStream();

// 创建一个byte类型的buffer字节数组，用于存放读取的本地文件

byte buffer[] = new byte[4 \* 1024];

int temp = 0;

// 循环读取文件

while ((temp = inputStream.read(buffer)) != -1) {

// 把数据写入到OuputStream对象中

outputStream.write(buffer, 0, temp);

}

// 发送读取的数据到服务端

outputStream.flush();

/\*\* 或创建一个报文，使用BufferedWriter写入,看你的需求 \*\*/ //String socketData = "[2143213;21343fjks;213]"; //BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(// socket.getOutputStream())); // writer.write(socketData.replace("\n", " ") + "\n"); // writer.flush(); /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

服务器端简单实现：

public void ServerReceviedByTcp() {

// 声明一个ServerSocket对象

ServerSocket serverSocket = null;

try {

// 创建一个ServerSocket对象，并让这个Socket在1989端口监听

serverSocket = new ServerSocket(1989);

// 调用ServerSocket的accept()方法，接受客户端所发送的请求，

// 如果客户端没有发送数据，那么该线程就停滞不继续

Socket socket = serverSocket.accept();

// 从Socket当中得到InputStream对象

InputStream inputStream = socket.getInputStream();

byte buffer[] = new byte[1024 \* 4];

int temp = 0;

// 从InputStream当中读取客户端所发送的数据

while ((temp = inputStream.read(buffer)) != -1) {

System.out.println(new String(buffer, 0, temp));

}

serverSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

## 4.2使用UDP协议通信

客户端发送数据实现：

protected void connectServerWithUDPSocket() {

DatagramSocket socket;

try {

//创建DatagramSocket对象并指定一个端口号，注意，如果客户端需要接收服务器的返回数据,

//还需要使用这个端口号来receive，所以一定要记住

socket = new DatagramSocket(1985);

//使用InetAddress(Inet4Address).getByName把IP地址转换为网络地址

InetAddress serverAddress = InetAddress.getByName("192.168.1.32");

//Inet4Address serverAddress = (Inet4Address) Inet4Address.getByName("192.168.1.32");

String str = "[2143213;21343fjks;213]";//设置要发送的报文

byte data[] = str.getBytes();//把字符串str字符串转换为字节数组

//创建一个DatagramPacket对象，用于发送数据。

//参数一：要发送的数据 参数二：数据的长度 参数三：服务端的网络地址 参数四：服务器端端口号

DatagramPacket packet = new DatagramPacket(data, data.length ,serverAddress ,10025);

socket.send(packet);//把数据发送到服务端。

} catch (SocketException e) {

e.printStackTrace();

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

客户端接收服务器返回的数据：

public void ReceiveServerSocketData() {

DatagramSocket socket;

try {

//实例化的端口号要和发送时的socket一致，否则收不到data

socket = new DatagramSocket(1985);

byte data[] = new byte[4 \* 1024];

//参数一:要接受的data 参数二：data的长度

DatagramPacket packet = new DatagramPacket(data, data.length);

socket.receive(packet);

//把接收到的data转换为String字符串

String result = new String(packet.getData(), packet.getOffset(),

packet.getLength());

socket.close();//不使用了记得要关闭

System.out.println("the number of reveived Socket is :" + flag

+ "udpData:" + result);

} catch (SocketException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

服务器接收客户端实现：

public void ServerReceviedByUdp(){

//创建一个DatagramSocket对象，并指定监听端口。（UDP使用DatagramSocket） DatagramSocket socket;

try {

socket = new DatagramSocket(10025);

//创建一个byte类型的数组，用于存放接收到得数据

byte data[] = new byte[4\*1024];

//创建一个DatagramPacket对象，并指定DatagramPacket对象的大小

DatagramPacket packet = new DatagramPacket(data,data.length);

//读取接收到得数据 socket.receive(packet);

//把客户端发送的数据转换为字符串。

//使用三个参数的String方法。参数一：数据包 参数二：起始位置 参数三：数据包长

String result = new String(packet.getData(),packet.getOffset() ,packet.getLength());

} catch (SocketException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

# 五、总结：

使用UDP方式android端和服务器端接收可以看出，其实android端和服务器端的发送和接收大庭相径，只要端口号正确了，相互通信就没有问题，TCP使用的是流的方式发送，UDP是以包的形式发送。